Statistical Real-Time Communication over Ethernet for Manufacturing Automation Systems

Seok-Kyu Kweon and Kang G. Shin
Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122
{skkweon,kgshin}@eecs.umich.edu

Qin Zheng
Argon Networks, Inc.
25 Porter Road
Littleton, MA 01460
zheng@argon.com

ABSTRACT

In order to realize real-time communication over Ethernet or fast Ethernet, one must be able to bound the medium access time within an acceptable limit. The multiple access nature of an Ethernet makes it impossible to guarantee a deterministic medium access time (hence packet delivery deadlines) to individual stations. However, one can bound the medium access time *statistically* by limiting the packet arrival rate at the medium access control (MAC) layer. While focusing on automated manufacturing systems as the main application, this paper considers the connection admission control (CAC) problem for statistically bounding the medium access time of Ethernet. Specifically, a packet is guaranteed to have a medium access time smaller than a predefined bound with a certain probability if the instantaneous packet arrival rate is kept below a certain threshold. Through an analysis, we first derived such a threshold. In order to keep the packet arrival rate under the given threshold, we employ a middleware which (i) resides between the transport layer and Ethernet datalink layer and (ii) smooths packet streams between them. The implementation of this middleware requires only a minimal change in the OS kernel *without* any modification to the current standard of Ethernet MAC protocol or TCP or UDP/IP stack. In order to solve the CAC problem, we derived the probability of transmitting a packet successfully upon each trial by modeling the MAC protocol, 1-persistent CSMA/CD, and the collision resolution protocol, Binary Exponential Backoff, of Ethernet. Using a simulation study, we have shown this analytic model to provide a reasonably accurate estimate of packet-loss (or deadline-miss) rate over Fast Ethernet. Finally, we implemented the middleware on the Linux OS, experimentally demonstrating the effectiveness of our approach in providing real-time communication over Ethernet.

Index Terms — 1-persistent CSMA/CD, (Fast) Ethernet, Binary Exponential Backoff, real-time communication, statistical guarantees.

The work reported in this paper was supported in part by Mitsubishi Electric Research Laboratory, Cambridge, MA, General Motors, Warren, MI, and by the Office of Naval Research under Grant N00014-94-1-0229.

1 Introduction

Advances in high-speed network technology have made it possible to transport traffic generated by various new applications over networks, resulting in an explosive growth of the Internet. Newly-emerging internet applications such as real-time audio/video communication require networks to have new functionalities that conventional packet-switched networks do not support. In particular, many of these applications require some form of Quality-of-Service (QoS) guarantees. For example, the manufacturing automation industry has been pursuing the use of commercial off-the-shelf (COTS) network products in communicating control messages between PLCs (Programmable Logic Controllers). Traditionally, proprietary networks such as Allen-Bradley's RIO (Remote Input/Output) Network and Control Net have been used for automated factory networks to meet the applications' stringent QoS requirements and deal with harsh working environments. However, the low price and the proven stability of COTS networks have made them attractive for automated manufacturing. Although various high-speed networks like ATM and FDDI are currently available, Ethernet has been drawing a special interest because of its extremely low price, maturity, and stability achieved through its wide deployment & acceptance. Despite its popularity and low-cost, Ethernet, however, has a serious drawback as a control message network. Its contention-based medium access control (MAC) protocol makes it impossible to provide predictable medium access time to component stations. Unlike other LANs (*e.g.*, FDDI or FieldBus), Ethernet has no mechanism to control the medium/channel access time. Although a switched Ethernet can provide deterministic delay guarantees by eliminating the possibility of packet collisions, its higher price has been slowing down its deployment in the manufacturing industry.

In this paper, we show the feasibility of using a generic Ethernet for transporting real-time control messages in an automated factory. We first analyze the medium access delay characteristics of Ethernet. Deriving a deterministic bound on the medium access time of Ethernet is generally intractable. If the packet arrival pattern is given in a deterministic form, such a bound may be derived, but it is of little value since the delay bound derived from the worst-case traffic arrival scenario is extremely large even under a very lightly-loaded condition. According to the 1-persistent CSMA/CD MAC protocol, those packets that have experienced a collision may experience more collisions. In the worst case, once two packets collide with each other, they will collide 15 more times before being discarded. So, the delay bound derived using the worst-case analysis is too large to be useful. We would therefore like to derive a practically meaningful (i.e., statistical) bound on the channel access time of Ethernet under several reasonable assumptions on input traffic and the functions of its MAC protocol. Specifically, we analytically derive a relationship between the statistical bound and the corresponding allowed input level. Through this analysis, one can obtain CAC for realizing statistical real-time communication over Ethernet. Our analysis considers the 1-persistent CSMA/CD MAC protocol with the Binary Exponential Backoff (BEB) strategy which is currently used in Ethernet.

Our analysis is based on the assumption that packets do not arrive in bursts and that the maximum packet arrival rate is limited through rate control or traffic smoothing. To enforce such a packet arrival behavior, we implemented the traffic smoothing function in the protocol stack. To minimize the change of the current network protocol standard, we developed a middleware that performs the traffic smoothing function while sitting between the transport and MAC layers.

Most of earlier work in the area of supporting real-time communication over Ethernet focused on modifying the Ethernet MAC sub-layer so that a bounded channel access time may be achieved [2–4]. These approaches are quite costly compared to using the well-established and widely-used current Ethernet standard. Venkatramani and Chiueh [5] proposed implementation of a virtual token ring over Ethernet in order to avoid packet collision. Token management is executed by a higher-layer protocol rather than the MAC, and thus their approach does not require modifying the hardware but the software. Specifically, it requires modifying the OS kernel, which is not simple. In addition, token management requires a number of functionalities, *e.g.*, restoration of a lost token, and it may overload the OS. Although our approach also requires modification of the kernel, the smoothing function is simple to implement, and hence, the changes are minimal. Another way to achieve a bounded channel access time is to use full duplex Ethernet switches instead of ordinary shared Ethernet hubs. This approach is also expensive compared to using shared Ethernet hubs. Especially, it is not economical to assign a port of an Ethernet switch to each individual control station in an automated factory network since the traffic-arrival rate of each station is quite low compared to link capacity. In most cases, an Ethernet switch is likely to be used to partition a large-scale LAN into multiple sub-LANs each of which consists of a shared Ethernet. In this environment, one must still be able to control the traffic arrival behavior of each sub-LAN in order to achieve end-to-end delay guarantees. Our approach can be used for this as well.

Much work has been done on the analysis of the CSMA/CD protocol. Assuming that the packet arrival is a Poisson process, researchers derived throughput and average packet delay. However, the throughput and average delay analysis is not enough nor acceptable in realizing real-time communication. Realization of real-time communication requires the knowledge of the tail distribution of packet delay rather than such average performance parameters. Beuerman and Coyle [6] derived the tail distribution of packet delay in a non-persistent CSMA/CD protocol, assuming that the average waiting time for retransmission is sufficiently larger than packet transmission time. However, the characteristic of the 1-persistent CSMA/CD protocol combined with BEB is quite different from their result on the non-persistent version.

Although the target application of our work is automated factory networks, providing delay guarantees in an Ethernet has broad impacts on many other areas. For example, in the global internetworking environment like the Internet, most end users are connected to the global network via LANs, and the most popular LAN standard is Ethernet. In this environment, controlling delays at Ethernet is the first step toward achieving the end-to-end QoS guarantees. If both LANs through which a source node and a destination node are connected, respectively, provide bounded local delays using our middleware and if the backbone network can provide delay guarantees, we can achieve a complete end-to-end delay guarantee between the source and destination pair.

The rest of the paper is organized as follows. Section 2 states the problem of providing statistical real-time guarantees over Ethernet. Section 3 derives the tail distribution of packet delay over Ethernet, and Section 4 demonstrates the effectiveness of the analysis through simulations. Section 5 presents experimental results. The paper concludes with Section 6.

2 Problem Statement

We first review some preliminary concepts necessary to state our main problem and discuss a software architecture for realizing statistical real-time communication over Ethernet.

A *deterministic* real-time channel is defined as a unidirectional virtual connection between two end-points that provides *a priori* deterministic or absolute guarantees for the timely delivery of packets in general packet-switching networks. On the other hand, a *statistical* real-time channel is defined as a unidirectional virtual circuit that guarantees the timely delivery of packets in statistical terms, i.e., the probability that a packet is lost during its transmission or misses its delivery deadline is less than a certain loss tolerance, Z [7]:

$$Pr(\text{end-to-end packet loss}) \leq Z, \tag{2.1}$$

or

$$Pr(\text{packet delay} > \text{delay bound}) \leq Z. \tag{2.2}$$

Unlike WAN or other LANs such as FDDI and FieldBus, Ethernet cannot control the medium access time of individual stations or connections. As a result, (1) only one type of QoS is provided to the entire component stations and all the connections established over the network, and (2) only one statistical real-time channel — shared by all stations — can be supported over a single Ethernet. For this reason, instead of using the above definition of a statistical real-time channel which was defined for general packet-switching networks, we introduce a new definition of a statistical real-time channel running over an Ethernet. In an Ethernet, if a newly-arrived packet results in a collision during its transmission, the transmission is stopped and the packet is rescheduled for transmission after some random delay. Hence, the delay that a packet experiences depends on the number of trials until its successful transmission. We therefore use the number of trials taken for a packet until its successful transmission as a performance measure in place of packet delay. A packet of a statistical real-time channel over the Ethernet must satisfy the following condition:

$$Pr(n \leq K) > 1 - Z, \tag{2.3}$$

where n is the number of trials taken to transmit the packet successfully. The counting starts when the packet arrives at the network from the application program. By relating the number of trials to the delay that a packet experiences before its successful transmission, we can easily transform Eq. (2.3) to a delay value. Let D_K^* be the worst-case delay of a packet when its transmission has succeeded at its K_{th} trial. Then, the following condition

$$Pr(D \leq D_K^*) > 1 - Z \tag{2.4}$$

is guaranteed to hold according to Eq. (2.3).

We will show a sufficient condition that Eq. (2.3) holds in the next section. The condition is that the total arrival rate of newly-generated packets from the component stations must be kept under a certain threshold which is called the *network-wide input limit*. Since the Ethernet MAC protocol is a fully-distributed one, it is impossible for each component station to know what the current instantaneous network-wide packet arrival rate is. What each component station only can do is to regulate its own packet arrival rate so that the network-wide input limit may not be violated. To this end, a portion of the network-wide input limit — called the *station-wide input limit* — is assigned to each component station. Each station regulates its packet stream which arrives from application programs so that the packet arrival rate at the MAC layer is kept under the station-wide input limit. With the cooperation of all the component stations, we can make sure that the network-wide input limit is not violated. Traffic smoothing is needed especially when packets arrive in a burst. Packets are more likely to collide when they arrive in a burst than sporadically. Moreover, smoothing packet streams will enable the packet arrival process to be modeled as a Poisson process, which is a basic assumption used in our analysis in the next section.

Traffic regulation or traffic smoothing is performed by a middleware at each component station. If packets arrived in a burst, the middleware buffers and sends them in such a way that their arrival times at the MAC layer are randomized while keeping the arrival rate under the station-wide input limit. Figure 1 shows the software architecture employed in our approach for realizing statistical real-time communication over Ethernet. In order to minimize the required changes in the current Ethernet standard, we insert the middleware between the transport (TCP/IP) layer and the Ethernet datalink layer.

In the next section, we derive the network-wide input limit for given loss tolerance and delay bound.

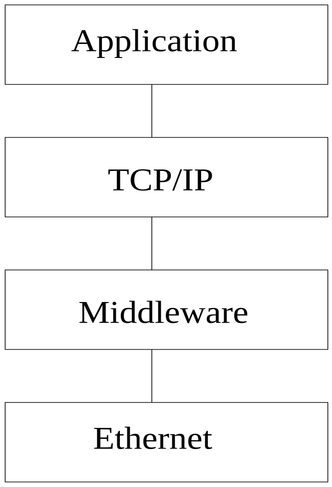


Figure 1: The software architecture

3 Analysis of 1-Persistent CSMA-CD Protocol

We model the MAC protocol of Ethernet, 1-persistent CSMA/CD protocol using a Semi-Markov process.

3.1 Modeling 1-persistent CSMA/CD networks

As we argued in Section 2, traffic smoothing makes packet arrival pattern amenable to the analysis and control of packet delays over Ethernet. So, the packet arrival process is assumed to form a Poisson process, which is not true in general LANs. Our analysis is based on the following assumptions.

A1. The traffic source of Ethernet consists of an infinite number of users who collectively form an independent Poisson source with an aggregate mean packet-generation rate of λ packets/sec. Moreover, the arrival process formed by the times at which packets are scheduled for transmission or retransmission, called the *channel offered traffic*, is assumed to be generated from another infinite population by a Poisson process with a time-varying parameter μ, which we call the *channel offered traffic rate*. Obviously, $\mu \geq \lambda$. This infinite population model assumes that each station has at most one packet requiring transmission at any time. Therefore, we ignore the queueing delay at each station in our analysis. We employed a time-varying parameter for the channel offered traffic rate in order to handle the bursty traffic characteristics of Ethernet.

A2. The channel reaches the steady state when the throughput remains at a constant (steady-state) value. Let each packet be of constant length requiring T seconds to transmit, and let $S = \lambda T$. Then, S is the average number of packets generated per packet transmission time. Under the steady-state condition in which the traffic arrival rate is equal to the traffic output rate, S can also be referred to as throughput.

A3. The new packet-arrival rate is sufficiently low compared to the network's transmission capacity (10 Mbps for Ethernet and 100 Mbps for fast Ethernet). This assumption is unavoidable to realize real-time communication over Ethernet. We will later determine how low the arrival rate should be. If the arrival rate is not sufficiently low, a portion of packets will experience collisions several times and hence large delays. This low arrival rate assumption will make the probability of a packet encountering multiple collisions negligible small.

A4. When a collision occurs, there are only two packets involved in that collision. In case of a low arrival rate, this is reasonable to assume. Also, we have to know the number of packets involved in a collision in order to derive the conditional collision probability when packets are retransmitted.

A5. The propagation delay between any two stations is equal to a constant, a. This is realistic for a 10 Base-T network which employs a star-topology wiring. For other topologies like a bus, one must set a to the largest propagation delay between any pair of stations. Since the collision probability increases as the propagation delay increases, choosing the largest propagation delay as a network parameter enables us to derive the worst-case (or an upper bound of) performance parameters.

Before describing our model, let's examine the operation of BEB briefly. When a packet collides with another packet, BEB sets the backoff time for the packet indicating when to try its retransmission. The backoff time is randomly chosen from $[0, 1, \cdots, 2^{n-1}] \times$ *slot_time*, where n is the number of times the packet has collided with other packets and *slot_time* is 512 bit times (in Ethernet, 51.2 μsecs, and in Fast Ethernet, 5.12 μsecs). Since the range of backoff time increases with the number of times a packet collided with other packets, the packet-arrival rate due to retransmissions is high when packets experience collisions a large number of times.

Based on the above assumptions and observations, we model a 1-persistent CSMA/CD network as a semi-Markov process (SMP)[1]: the SMP has two types of operating modes, depending on the packet-arrival rate: QUIET and BURST. These two operating modes are introduced to reflect the bursty nature of traffic arrival on Ethernet. In QUIET mode, the arrival rate of retransmitted packets is low, as compared to the arrival rate of new packets. This happens when retransmitted packets have already experienced collisions a number of times, so their backoff times are quite large. According to A1, in this mode, the arrival process of both new packets and retransmitted packets forms a Poisson process with rate g. Upon occurrence of a collision, the system goes into BURST mode. In this mode, the channel offered traffic rate, β, is dominated by the arrival rate of retransmitted packets, and the collision probability is much higher than in QUIET mode. In an Ethernet where the arrival rate of new packets is quite low, if no collision occurs for a sufficiently long time, the channel offered traffic rate is

[1] this modeling draws on Vo-Dai's work [8] which we modified to accommodate the bursty nature of Ethernet traffic

very small and close to the arrival rate of new packets. However, once a collision occurs, the channel offered traffic rate increases abruptly due to the small backoff time. In this state, the probability of another collision when they are scheduled for retransmissions is very high. BURST mode is introduced to represent this situation. We can analyze the behavior of Ethernet in BURST mode by considering the worst-case scenario in terms of collisions. The worst-case scenario occurs when packets involved in a collision are newly-arrived. From A4, only two packets are involved in a collision at a time. In addition, we assume that the arrival process of both new and retransmitted packets become a Poisson process with rate β. Since the arrival rate of new packets is negligible compared to that of retransmitted packets, β is approximated by the arrival rate of retransmitted packets. Using these assumptions, we can determine β as follows. First, packets' backoff times are either 0 or 5.12 μsecs (51.2 μsecs) in Fast Ethernet (Ethernet)[2]since collided packets are all newly-arrived. Then, the probability that no packets will be scheduled during the first of two $slot\_time$ periods which are respectively $[0, 5.12)$ μsecs and $[5.12, 2 \times 5.12)$ μsecs, is 1/4. Thus, the average arrival rate in BURST mode, β, is obtained from the relation:

$$Pr(0 \text{ packets arrive during one } slot\_time) = 1/4.$$

Since we assumed that the arrival process is Poisson,

$$Pr(0 \text{ packets arrive during one } slot\_time) = e^{-\beta \cdot slot\_time},$$

and thus,

$$e^{-\beta \cdot slot\_time} = \frac{1}{4}.$$

Rearranging the terms,

$$\beta = \log 4/slot\_time \approx 1.3863/slot\_time.$$

Based on these two modes, we obtain the following SMP model for Ethernet with 9 states each of which belongs to either QUIET or BURST mode. (Figure 2 depicts each state.)

0: The QUIET-mode transmission state. A single station's transmission has lasted for the period of propagation delay, a, without interference from any other station and still continues, and the packet started transmission from the station in QUIET mode. Since all users are now aware of this transmission, they will not interfere with it.

1: The idle state. No station is transmitting packets on the channel. This state belongs to QUIET mode.

2: The QUIET-mode single contention state. One station started transmitting a packet in QUIET mode and continues transmission on the channel, but it has not been heard by all of the other

[2]From now and on, we use fast Ethernet in all examples in this section

stations. We assume that this state continues until the first transmission is heard by all the stations whether or not other stations transmit packets. Thus, the sojourn time of state 2 is the propagation delay, a.

3: The BURST-mode collision state. Two or more stations have been transmitting simultaneously and the first transmission was heard by all the stations. After sensing the collision, the stations stop transmitting their packets and transmit a jamming signal.

4: The QUIET-mode multi-contention state. Two or more stations start transmitting packets at exactly the same time in QUIET mode but their transmissions have not yet been heard by each other, or by others. This happens only when two or more stations schedule their packets for transmission in the same QUIET-mode transmission state (state 0), and thus, start transmitting packets as soon as the channel is free.

5: The BURST-mode singular collision state. This state is subordinate to states 4 and 7, and is entered from states 4 and 7 only when no other packets, except those which originally initiated states 4 and 7, are being transmitted.

6: The BURST-mode single contention state. One station started transmitting a packet in BURST mode and continues transmitting on the channel, but it has not been heard by all of the other stations. We assume that, as in state 2, this state continues until the first transmission is heard by all the stations whether or not other stations transmit. Therefore, the sojourn time of state 6 is the propagation delay, a, as in state 2.

7: The BURST-mode multi-contention state. Two or more stations start transmitting packets at exactly the same time in BURST mode but their transmissions have not yet been heard by each other or by others. This happens only when two or more stations schedule their packets for transmission in the same BURST-mode transmission state (state 8), the same BURST-mode collision state (state 3), or the same BURST-mode singular collision state (state 5) (and thus, start transmitting packets as soon as the channel becomes free).

8: The BURST-mode transmission state. A single station's transmission time has lasted for the propagation delay, a, without interference from any other station and the packet has arrived during BURST mode. Since all users are now aware of this transmission, they will not interfere with it.

At any time, the channel can be in one of these 9 states. In this model, the channel starts in state 1 in QUIET mode. Upon occurrence of a collision, the channel enters, and stays in, BURST mode until the channel becomes idle.

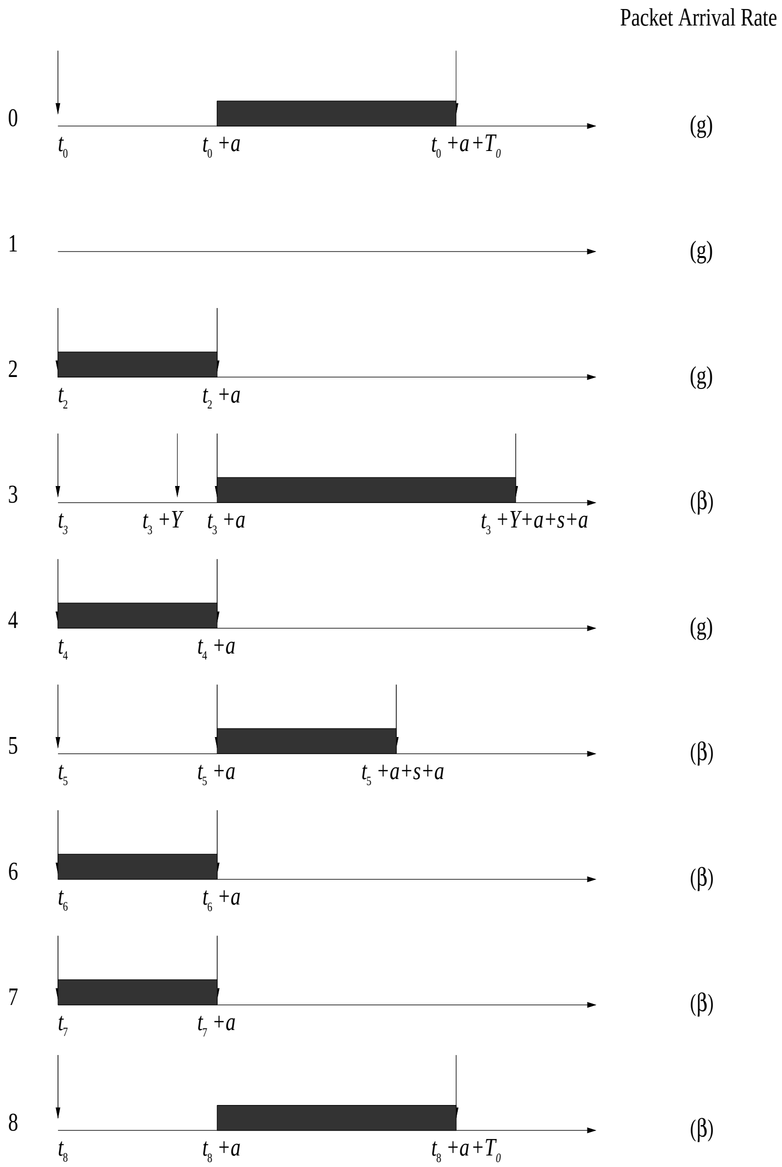


Figure 2: Channel states

For the embedded Markov chain of the SMP, the following steady-state equation must hold:

$$\pi = \pi M, \tag{3.1}$$

where $\pi = (\pi_0 \ \pi_2 \ \cdots \ \pi_8)$, π_i's are steady-state probabilities, and M is the state-transition matrix given by

$$M = \begin{pmatrix} 0 & A_1 & A_2 & 0 & 1-A_1-A_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ e^{-ga} & 0 & 0 & 1-e^{-ga} & 0 & 0 & 0 & 0 & 0 \\ 0 & A_3 & 0 & 0 & 0 & 0 & A_4 & 1-A_3-A_4 & 0 \\ 0 & 0 & 0 & 1-e^{-ga} & 0 & e^{-ga} & 0 & 0 & 0 \\ 0 & A_5 & 0 & 0 & 0 & 0 & A_6 & 1-A_5-A_6 & 0 \\ 0 & 0 & 0 & 1-e^{-\beta a} & 0 & 0 & 0 & 0 & e^{-\beta a} \\ 0 & 0 & 0 & 1-e^{-\beta a} & 0 & e^{-\beta a} & 0 & 0 & 0 \\ 0 & A_7 & 0 & 0 & 0 & 0 & A_8 & 1-A_7-A_8 & 0 \end{pmatrix}$$

where

$$A_1 = \int_0^\infty e^{-gx} dF(x),$$

$$A_2 = \int_0^\infty gxe^{-gx} dF(x),$$

$$A_3 = \int_0^\infty e^{-\beta(a+s+x)} dG(x),$$

$$A_4 = \int_0^\infty \beta(a+s+x)e^{-\beta(a+s+x)} dG(x),$$

$$A_5 = e^{-\beta(s+a)},$$

$$A_6 = \beta(s+a)e^{-\beta(s+a)},$$

$$A_7 = \int_0^\infty e^{-\beta x} dF(x),$$

and

$$A_8 = \int_0^\infty \beta xe^{-\beta x} dF(x).$$

s is the jamming time[3]; $F(x)$ is the probability distribution function of packet-transmission times; and $G(x)$ is the probability distribution function of the latest transmission initiated within a contention period. In state 3 of Figure 2, $t_3 + Y$ indicates the latest transmission initiated within the contention period. See [8] for a detailed account of derivation of $G(x)$. Note that $G(x)$ depends on the packet-arrival rate in both contention states 2 and 6. Considering that the arrival rates for states 2 and 6 are

[3]We merged the interframe gap into the jamming time and the packet-transmission time, respectively.

g and β, respectively, let $G(x;g)$ and $G(x;\beta)$ denote the probability distribution functions of the latest transmission initiated in state 2 and state 6, respectively. Then, it is easy to show

$$G(x;g) \geq G(x;\beta), \ \ \forall \ x.$$

That is, $Y(\beta)$ is *probabilistically larger than* $Y(g)$ [9], where $Y(g)$ and $Y(\beta)$ are the latest transmissions initiated in states 2 and 6 in reference to their starting times, respectively. Then, $G(x)$ is given as the weighted sum of $G(x;g)$ and $G(x;\beta)$. However, since one needs to consider the worst case in terms of packet loss, we choose $G(x;\beta)$ to be $G(x)$. Thus, we obtain

$$G(x) = \begin{cases} 0 & x < 0 \\ (e^{\beta x} - 1)e^{-a\beta}(1 - e^{-a\beta}) & 0 \leq x < a. \\ 1 & x \geq a \end{cases}$$

The transition probability is derived using the Poisson arrival assumption. For example, M_{02} given by A_2 is obtained by considering the fact that there must be only one packet arrival during one packet-transmission time, in order for the system to go from state 0 to state 1. Other elements of M are obtained in a similar way.

The steady-state probabilities, $\pi_i, i = 0, 1, \cdots, 8$ are a solution to both Eq. (3.1) and the following equation:

$$\sum_{j=1}^{8} \pi_j = 1.$$

Here we omit the algebraic solution for this system of equations.

Let T_i denote the time spent in each state of the SMP. Then, the expected time spent in each state of the SMP, $E[T_i]$, is given by

$$\begin{aligned} E[T_0] &= \int_0^\infty x dF(x), \\ E[T_1] &= \frac{1}{g}, \\ E[T_2] &= a, \\ E[T_3] &= a + s + \int_0^a x dG(x), \\ E[T_4] &= a, \\ E[T_5] &= a + s, \\ E[T_6] &= E[T_2], \\ E[T_7] &= E[T_4], \end{aligned} \tag{3.2}$$

and

$$E[T_8] = E[T_0].$$

Eq. (3.2) is derived from the assumption that the arrival process is Poisson with rate g.

Based on the steady-state probabilities and the expected sojourn time in each state, we can derive the throughput S which is defined as the fraction of time spent on actual transmissions:

$$S = \frac{\pi_0 E[T_0] + \pi_8 E[T_8]}{\sum_{j=0}^{8} \pi_j E[T_j]}.$$

From the steady-state assumption, S can be considered as the input rate due to newly-arriving traffic at all stations.

Now, in order to derive the probability of a packet being transmitted successfully at its first trial, we insert a *probe* packet into this system. First, we calculate the probability of the probe packet finding the system in state i upon its arrival at the system as:

$$P_i = \frac{\pi_i E[T_i]}{\sum_{j=0}^{8} \pi_j E[T_j]}, \qquad i = 0, \cdots, 8.$$

This is based on the assumption that the new arrivals follow a Poisson process, and thus, that the probe packet's arrival time is uniformly-distributed in time.

Next, for each state, we can calculate the probability of the probe packet being transmitted successfully in that state as:

$$u_0 = \int_0^\infty e^{-gx} dF(x) \cdot e^{-ga}, \tag{3.3}$$

$$u_1 = e^{-ga},$$

$$u_2 = 0,$$

$$u_3 = \int_0^\infty e^{-\beta(a+s+x)} dG(x) \cdot e^{-\beta a},$$

$$u_4 = 0,$$

$$u_5 = e^{-\beta(s+a)} \cdot e^{-\beta a},$$

$$u_6 = 0,$$

$$u_7 = 0,$$

and

$$u_8 = \int_0^\infty e^{-\beta x} dF(x) \cdot e^{-\beta a}.$$

u_0 is obtained as follows. When the probe packet has arrived at the system which is in state 0, i.e., the QUIET-mode transmission state, it can be successfully transmitted only if there are no packet arrivals during the on-going packet-transmission time and no packet arrival during the probe packet's contention period, $[0, a]$. The probabilities of these events are given by $\int_0^\infty e^{-gx} dF(x)$ and e^{-ga}, respectively. Thus, we obtain u_0; similarly, other u_i's are obtained.

Finally, the probability of the probe packet being transmitted successfully at its first trial is given as a weighted sum:

$$P_s = \sum_{j=0}^{8} u_j P_j.$$

3.2 Calculating Success Probability upon Retransmission

Although we employed a Poisson process for packet arrivals in which a packet's waiting time until its transmission is independent of the number of trials, the collision probability of a packet depends heavily on the number of collisions that the packet has experienced, as well as the channel offered traffic rates, g and β, in Ethernet, because of its BEB strategy. It makes the probability of a packet's successful transmission dependent on the number of (re)transmission trials. So, a packet's success probability at its second trial is different from that at its first trial, and each trial for retransmission has a different success probability.

We approach this problem by considering the conditional probability that the probe packet is transmitted successfully at its second trial, given that its first trial has failed. In this case, the BEB strategy makes the station choose one of two backoff times, $0 \times slot\_time$ and $1 \times slot\_time$. In order to calculate the probability of collision at the second trial, we need information about the number of packets which have been involved in the first collision. As we did when determining β, we assume that the number of packets involved in the first collision is 2, including the probe packet itself. As we argued earlier, the network load condition in which real-time communication can be provided will be lightly-loaded. Thus, we expect the above assumption to be valid. Under this assumption, the two packets involved in the first collision must choose different backoff times in order for the second trial to be successful. Let the probe packet choose the first backoff time 0 and the other packet choose the second backoff time, $slot\_time$. The probability of this event is 1/4, and the probability of the probe packet being successfully transmitted in this event, P_{21}, is given by

$$\begin{aligned} P_{21} &= Pr(\text{no arrivals during the collision period}) \times \\ &\quad Pr(\text{no arrivals during the contention period of the probe packet}). \end{aligned} \tag{3.4}$$

Since we are dealing with collided packets separately from the other packet arrivals, the channel offered traffic rate in BURST mode is not β but g in this calculation, because the number of packets which were involved in the collision is assumed to be 2. Then, the first term in Eq. (3.4) is given by

$$Pr(\text{no arrivals during the collision period}) = e^{-g(s+2a)},$$

The length of the collision period is the sojourn time of state 3, and we take its maximum value, $s+2a$, in order to obtain the worst-case success probability. The second term is given by

$$Pr(\text{no arrivals during the contention period of the probe packet}) = e^{-ga}.$$

Thus,

$$P_{21} = e^{-g(s+3a)}.$$

Next, we consider P_{22}, the success probability when the probe packet chooses the second backoff time. Since the other packet has chosen the first backoff time, the probe packet must wait for the other packet to finish its transmission if it does not collide with a third packet. Otherwise, the situation gets more complicated. That is, when the other packet has collided with a third packet, we have too many possibilities of successfully transmitting the probe packet. In order to avoid such complexity, we simply set the success probability of the probe packet to zero in that case, which is the worst-case lower bound of the success probability. This affects our analysis very little since the probability that the other packet collides with a third packet is very small. Then,

$$\begin{aligned} P_{22} &= Pr(\text{no arrivals during the collision period}) \times \\ &\quad Pr(\text{no arrivals during the contention period of the other packet}) \times \\ &\quad Pr(\text{no arrivals during the transmission period of the other packet}) \times \\ &\quad Pr(\text{no arrivals during the contention period of the probe packet}). \end{aligned}$$

Substituting all the terms,

$$P_{22} = e^{-g(s+2a)} \cdot e^{-ga} \cdot \int_0^\infty e^{-gx} dF(x) \cdot e^{-ga}.$$

Then, the conditional probability that the probe packet is transmitted successfully at its second trial given that its first trial has failed, $P_{2|1}$, is given by

$$\begin{aligned} P_{2|1} &= \frac{1}{4} P_{21} + \frac{1}{4} P_{22} \\ &= \frac{1}{4}\{e^{-g(s+3a)} + e^{-g(s+4a)} \cdot \int_0^\infty e^{-gx} dF(x)\}. \end{aligned}$$

Similarly, one can approximate the conditional probability of the probe packet being transmitted successfully at the third or later trials given that it has failed at its previous trials. However, since we have already employed many assumptions in obtaining the conditional probability for the second trial, such approximations will add larger and larger errors as the number of trials increases. Thus, we use the conditional probability for the second trial as the estimate for the conditional probability for later trials instead of approximating them. Because of its smallest backoff time, the second trial provides the worst-case situation in terms of packet-arrival pattern for later trials. Thus, $P_{2|1}$ can be used as the worst-case estimate for later trials' conditional success probabilities. Thus, for $n > 2$, we set $P_{n|n-1} := P_{2|1}$.

Finally, we can obtain the probability of a packet being transmitted successfully within K trials using the conditional success probabilities. It is given in the following recursive form:

$$P_K = Pr(n \le K) = P_{K-1} + (1 - P_{K-1}) P_{K|K-1},$$

where $P_0 = 0$ and $P_{1|0} = P_s$.

In our approach, a packet which did not get transmitted within K trials is considered missing its delivery deadline, and thus, lost. In this case, the delivery deadline of a packet is given as the delay that a packet which is transmitted successfully in its K_{th} trial experiences in the worst scenario possible. Such a packet experiences the maximum delay possible when it experienced the longest waiting times and the longest backoff times until the $(K-1)_{th}$ trial in all its previous trials and finally succeeded in the K_{th} trial. Again, we pick such a packet as a probe packet. Let's derive this worst-case delay that the probe packet experiences when it succeeded in its K_{th} trial. The worst case, the probe packet being delayed before being scheduled for its first transmission trial, happens when the probe packet arrives for transmission at the system which is at the beginning of state 0. In this case, the probe packet must wait for the in-progress packet to finish its transmission before being transmitted. Since the length of the transmission time of the in-progress packet is given as T_0, its maximum is given as $\max(T_0)$. Upon completion of the transmission of the in-progress packet, the probe packet starts transmission. If the probe packet collides with other packets during this transmission period, the transmission of the probe packet will be stopped, and this interrupted transmission time is counted toward the delay of the probe packet. We call it the *collision period*. The length of the collision period is given by the sum of the sojourn times of states 2 and 3, i.e., T_2 and T_3. Since we are considering the worst case, we take the maximum value of T_3, $s + 2a$, so the maximum collision period is $3a + s$. After experiencing its worst-case collision period, the probe packet will be scheduled for retransmission (for the second trial). Again, in the worst case, the probe packet will take the largest backoff time, $1 \times slot\_time$, instead of $0 \times slot\_time$. If the system has entered state 0 by another packet just before the probe packet is scheduled for transmission in its second trial, the probe packet must wait for the in-progress transmission to finish again. The probe packet can then finish its transmission unless another collision happens. Thus, $2 \cdot \max(T_0) + T_2 + \max(T_3) + slot\_time$ is the worst-case delay of the probe packet before being transmitted in its second trial. The worst-case delay of the probe packet when it succeeds in its K_{th} trial is obtained using a similar argument. Thus, the worst-case delay when a packet is successfully transmitted within K trials is given by

$$D^*(K) = K \cdot \max(T_0) + (K-1) \cdot \{T_2 + \max(T_3)\} + \sum_{j=1}^{K-1} (2^j - 1) \cdot slot\_time + \max(T_0) \tag{3.5}$$

The last term, $\max(T_0)$, is to consider the probe packet's worst-case transmission time. Arranging terms, we obtain

$$D^*(K) = (K+1) \cdot \max(T_0) + (K-1) \cdot (3a + s) + (2^K - K - 1) \cdot slot\_time. \tag{3.6}$$

4 Simulation Results

To validate our analytic model, we simulated a 100BASE-T network with the following parameters. The transmission capacity is 100 Mbits/sec and the propagation delay between any two stations is 1.7 μsec assuming that the collision domain diameter (maximum distance between stations) is 400 m. The slot time ("unit" time for backoff) is 512 bit times or 5.12 μsecs. The interframe gap ("gap" time) is 0.96 μsecs and the jamming time is 0.48 μsecs. The number of stations is set to 20. Both fixed- and variable-size packets were simulated. For the case of fixed-size packets, the packet size was set to 64, 500, 1000 and 1500 bytes. For the case of variable-size packets, a mixture of packets of 64 and 1500 bytes were simulated. In the variable-size case, the analytic results were more optimistic in estimating the packet-loss rate than the simulation results, due mainly to the violation of the infinite population assumption. That is, even under lightly-loaded conditions, packet queues were built up by shorter packets during the transmission of longer packets. (Note that the number of packets is much larger if the packet size gets smaller while keeping the same load level.) This violates our assumption that each station can hold at most one packet. As a result, our Ethernet model based on the SMP process is ineffective in deriving CAC for realizing real-time communication in the variable-size case. Here we only present the comparison results for the fixed-size case.

We simulated the model for several seconds to dozens of seconds for various traffic loads. The number of packets arrived from the stations is approximately 50,000 for each load considered. We count the number of trials until a newly-arrived packet succeeds in its transmission and calculate the conditional success probability for each trial and derive the packet-loss ratio within n trials where $n = 1, \cdots, 5$.

Figure 3 shows the analytic result of the throughput for varying the channel offered traffic rate, g when the packet size is set to 1500 bytes. Note that g is the rate of both new packet arrivals and retransmission trials in QUIET mode. Since we have two channel offered traffic rates, g and β in our model, Figure 3 must read differently from the results in [8, 10]. Figure 3 indicates the relation between g and the input load from outside the system. We determine g from Figure 3, given the input load, using the steady-state condition assumption as discussed in Section 3.2. Unlike g, β is a fixed parameter that is independent of the throughput.

Using the channel offered traffic rate g obtained from Figure 3, we calculated the conditional success probability for the first and second trials following the steps explained in Section 3.2, and plotted them in Figure 4 along with the simulation results when the packet size is set to 1500 bytes. The analytic results precisely match the simulation results for the first trial under lightly-loaded conditions which are of prime interest to the design of real-time communication[4]. For the second trial, the analytic results

[4] the 99% confidence interval was less than 1%

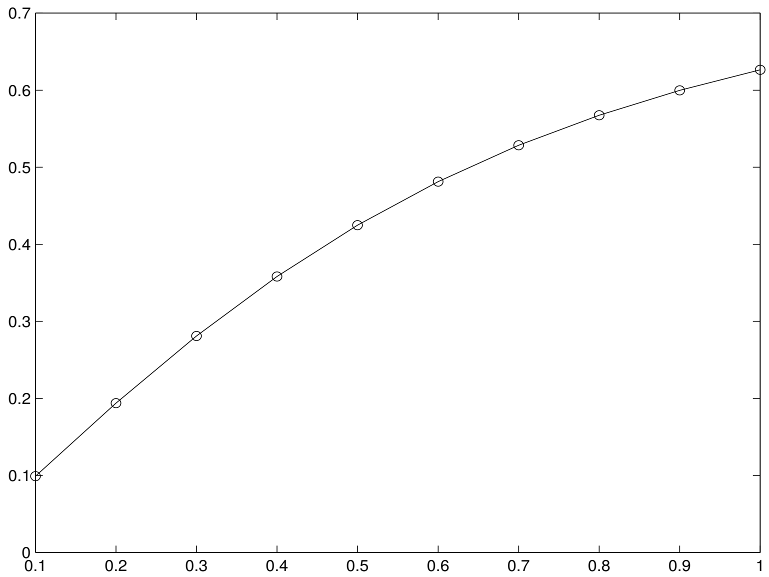


Figure 3: Throughput – 1500 bytes

provide a reasonable lower bound for the simulation results except for very lightly-loaded conditions, where the analytic results are more optimistic than the simulation results. For later trials, the estimate for the second trials is found to work as a good lower bound, which supports our argument in Section 3.2.

Figure 5 shows packet-loss ratios when a packet which did not succeed in its transmission within n trials is considered lost. We calculated the worst-case delay bound for each trial using Eq. (3.6). This is shown for different packet sizes in Table 1. From Figure 5, when the packet size is set to 1500 bytes,

packet size (bytes)	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$
64	10.24	25.18	48.74	89.55	164.8
500	80.00	131.4	192.8	274.1	395.5
1000	160.00	251.5	353.1	475.0	637.4
1500	240.0	371.5	513.2	675.3	878.0

Table 1: Delay bounds in μsecs

one can see that the input load must be kept under 30% in order to achieve the packet-loss ratio under 1% if the packet-delivery deadline is set to 878 μsecs. For more stringent packet-loss ratios, we must either increase the delay bound or decrease the input-load level.

Although we do not show the simulation results for the different packet sizes, they showed similar trends. The difference is that, as the packet size decreases, the performance gets worse. That is,

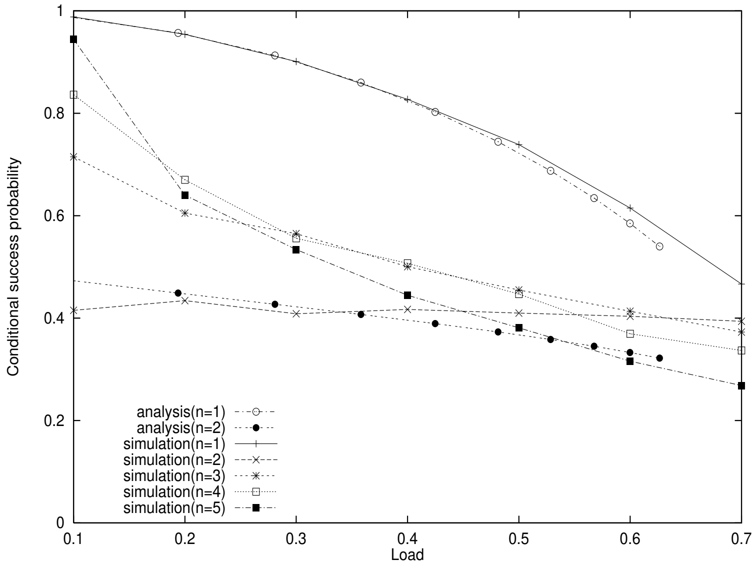


Figure 4: Conditional success probability – 1500 bytes

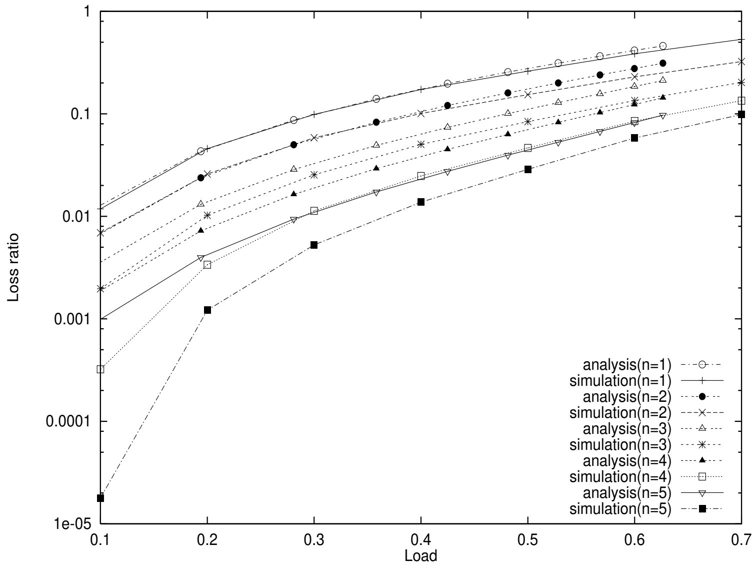


Figure 5: Loss ratio – 1500 bytes

network utilization must be lowered significantly in order to keep the same packet-loss tolerance for a given n. This is because packet-arrival rates are higher in smaller packet sizes under the same input load. Of course, the delay bound gets smaller in case of smaller packet sizes for the same trial number as shown in Table 1. From this result, we can conclude that, for realizing real-time communication over Ethernet with reasonable packet-loss ratios, choosing a larger packet size is more advantageous as long as this does not result in an unreasonably large delay bound.

5 Experimental Evaluation

To verify the results of both analytic modeling and simulation studies, we built a testbed and conducted experiments on it. The target application for our experiments is an automated factory. Traditionally, FieldBus-type networks have been used for transporting real-time control messages between PLCs in order to provide deterministic delay guarantees. In addition to a FieldBus, PLCs are interconnected via an Ethernet which also connects PLCs to Management Information Systems (MIS) and program development & support systems. In this environment, real-time control messages are exchanged between PLCs through the FieldBus, and non-real-time messages are exchanged via the Ethernet between each PLC and MIS applications and program development & support systems. We performed experiments on this testbed to investigate the possibility of integrating the current FieldBus – Ethernet combination into a single Ethernet. If the integrated system can still provide adequate real-time performance, one can eliminate the proprietary FieldBus from the automated factory network.

In the current factory networking environment, real-time control messages are transported through the FieldBus during the factory operation, and each control station generates at most one single maximum-sized (1500 bytes) IP datagram every 1–2 seconds or larger. That is, the real-time packet generation rate by each station is very low, e.g., in the range of 10 Kbps. During the factory operation, non-real-time traffic is generated irregularly by PLCs and MIS applications, mainly for the purpose of monitoring, and transported over the Ethernet. Since in most cases this non-real-time traffic is transferred via ftp, the instantaneous arrival rate of traffic generated by each station can be quite high, *e.g.*, 8–9 Mbps. In an integrated environment, real-time messages are transported over the same Ethernet along with this non-real-time traffic and are likely to experience collisions during the transmission of ftp traffic. Even though the long-term average network load is quite low, real-time messages can suffer a large delay because of such an instantaneous surge of network load. To provide delivery delay guarantees for control messages, it is important to keep the instantaneous network load under a pre-specified threshold, which we derived in Section 3. For this purpose, we designed and implemented a middleware which performs traffic smoothing in the Linux OS. The middleware resides between the TCP/IP layer and the Ethernet datalink layer and smooths the packet stream from the TCP/IP layer so that the instantaneous packet-arrival rate may be kept under a given station-wide input limit.

5.1 The Experimental Environment

The testbed consists of three 300 MHz Intel Pentium II PCs and one NEC 75 MHz laptop computer, and they are connected through a 5-port 10 BASE-T Ethernet hub. The collision domain diameter is 10 m. Our middleware is installed on all of the PCs, and smooths the packet stream using a leaky-bucket regulator [11] which has a token bucket. The token bucket depth (TBD) limits the maximum number of tokens that can be contained in the token bucket. Up to TBD tokens are added to the bucket every refresh period (RP), specified in seconds. If the number of tokens exceeds TBD, overflow tokens are discarded. When a packet arrives from the IP layer, the middleware forwards it to the Ethernet NIC (network interface card) after removing as many tokens as the size of the packet (in bytes) if at least one token exists in the token bucket. (When the number of available tokens is smaller than the packet size, tokens are allowed to be "borrowed." So, the number of tokens can be below zero.) Otherwise, the packet is held in the buffer until one or more tokens become available. By changing RP and TBD, one can control the burstiness of a packet stream while keeping the same average throughput guarantee. For example, if we set the ratio of TBD to RP to 312,500, the average throughput guaranteed for a station is 312.5 Kbytes/sec or 2.5 Mbps. Two possible pairs of (TBD, RP) satisfying the ratio are (1500, 0.0048) and (150000, 0.48). When (TBD, RP) is set to (1500, 0.0048), the maximum amount of traffic that can be transmitted consecutively is limited to 2999 bytes (1499 bytes plus 1500 bytes). This case smooths traffic on a very small time scale and generates a Poisson-like output. In the other case, up to 151,499 bytes can be transmitted consecutively. Although the average traffic arrival rate is the same, this case generates much burstier output and emulates the traffic arrival scenario of an ordinary Ethernet. The middleware described above can, in certain cases, generate a deterministic output pattern. Although we can make the middleware generate more Poisson-like traffic arrival patterns by randomizing RP, we used constant RPs in order not to burden the kernel. In the experiment, we avoided such a deterministic traffic arrival scenario by randomizing the traffic-generation time at the application layer.

To prevent real-time packets from being delayed due to the non-real-time packets generated within the same station, we give priority to real-time packets using the TOS (Type Of Service) field of IP header.

Three PCs (two 300 MHz Pentium II PCs and the NEC laptop) were used for generating non-real-time packets. The size of IP datagrams was set to the maximum or Ethernet's MTU, i.e., 1500 bytes. Thanks to the middleware, the maximum arrival rate of non-real-time packets is limited under a specified threshold. Using TBD and RP, we control the traffic arrival rate of each individual station. Non-real-time packets are used to drive the network under a given load. The remaining 300 MHz

Pentium II PC is used to generate real-time packets. It sends a real-time message in a maximum-sized (1500 bytes long) IP datagram to one of the Pentium II PCs, and the recipient echoes it back to the originator immediately. The originator measures the roundtrip delay of each real-time message. Since this roundtrip delay includes processing delays at high layers at both stations, it is larger than the roundtrip delay between Ethernet NICs. We, however, ignore processing delays in the higher layer and consider the half of the measured roundtrip delay of a real-time message medium access delay which is defined as the time a packet is successfully transmitted since its arrival at the Ethernet NIC at the source node. Therefore, the actual medium access delay is smaller than the measured one. Since the arrival rate of real-time packets at a single station is usually very small compared to the link capacity in an automated factory network, we keep the real-time traffic arrival rate at a small value. In our experiment, we kept the average real-time packet inter-arrival time at 0.3 seconds at each station, and thus, the arrival rate of a station is approximately 40 Kbps. The distribution of real-time packet inter-arrival time was uniform.

5.2 Results

We performed two sets of experiments for a range of network loads: 0.2, 0.3, 0.4 and 0.5. In the first set, we enforced traffic smoothing on a very small time scale. In this case, TBD was set to 1500, and any two maximum-sized packets were not allowed to be transmitted consecutively. In the second set, we enforced traffic smoothing on a long time scale. In this case, TBD was set to 1.5×10^5, and up to 100 maximum-sized packets were allowed to be transmitted consecutively. This setting was used to emulate the worst-case traffic arrival behavior of Ethernet without traffic smoothing. In our analysis we set the maximum number of transmission trials until the packet is declared lost, to 10. This imposed the delay bound to be 64.8 mseconds. So, in our measurement, we considered a packet lost if its roundtrip delay is larger than 129.6 mseconds.

Figure 6 plotted three graphs for the loss ratio of real-time packets. The one with the largest packet loss ratios represents the experimental results when a burst was allowed to be transmitted, i.e., TBD was set to 1.5×10^5. The one with the smallest packet loss ratios represents the result when strict traffic smoothing was enforced, i.e., TBD was set to 1500. The one with the moderate packet loss ratios represents the analysis result. The confidence interval of each measurement is 10^{-5}. Comparing the analysis and the measured result in the coarse time scale traffic smoothing mode, the measured packet loss ratios are much larger than the estimated ones. Considering that our analysis was based on the Poisson arrival assumption while the traffic pattern used in the experiment was bursty, this is an expected result. In the strict traffic smoothing mode, the measured packet loss ratios are upper-bounded by the estimated ones.

The difference between the two experiment settings is more pronounced in the roundtrip delay

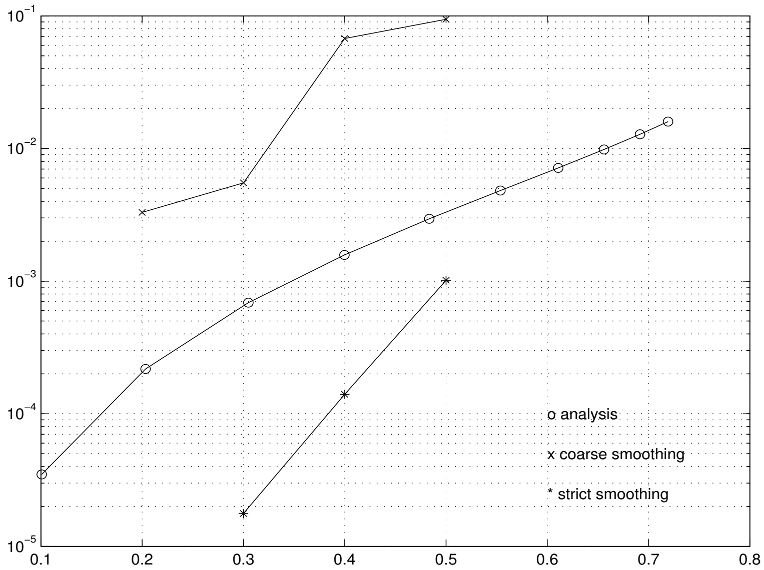
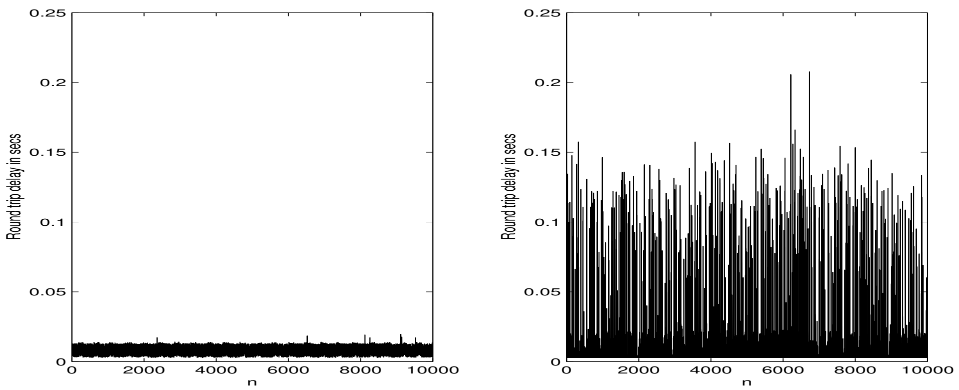


Figure 6: Loss ratio comparison

sequence of the entire real-time packets when the network load was set to 0.3 as shown in Figure 7 where n indicates the sequence number of each packet within the stream. The variability of roundtrip delay in the coarse time scale traffic smoothing mode shown in Figure 7(b) is much larger than that in the strict traffic smoothing mode shown in Figure 7(a). The average delays were 8.2 mseconds and 8.4 mseconds in the strict traffic smoothing mode and the coarse time scale traffic smoothing mode, respectively.



(a) (b)

Figure 7: Roundtrip delay sequence

6 Conclusion

In this paper, we presented a methodology for providing statistical delay guarantees over Ethernet (and/or Fast Ethernet). We analyzed the Ethernet MAC protocol using a semi-Markov process model and derived a network-wide input limit for achieving a target transmission success ratio. The network-wide input limit is kept by enforcing each component station to control its instantaneous traffic arrival rate under its station-wide input limit. To this end, we installed a middleware at each station. The middleware resides between the UDP or TCP/IP layer and Ethernet datalink layer and keeps the traffic arrival rate under the station-wide input limit by smoothing a bursty packet stream. We also showed the effectiveness of our analysis using a simulation study. In addition, we implemented the traffic smoothing middleware in the Linux OS, and conducted an empirical study. The experimental results support our analysis result and our approach to statistical real-time communication over Ethernet.

In our approach, each station is required to control its own traffic generation rate using a middleware. As the number of stations increases, the station-wide input limits must decrease accordingly. This is undesirable in an automated factory networking environment where non-real-time traffic is generated once in a while and thus the average load due to non-real-time traffic is quite low. When non-real-time traffic (like an ftp session) is generated by a station, it may experience a large delay because of a small station-wide input limit. To avoid this in a large-scale LAN, we may have to partition the network into multiple small sub-LANs and connect them through "intelligent" gateways or Ethernet switches. By intelligent, we mean that gateways or Ethernet switches must be able to manage network resources to provide end-to-end QoS guarantees to all the component stations. They are required to have functionalities like packet scheduling, connection admission control, and traffic smoothing. We are currently investigating several issues related to this matter. Specifically, we are looking at the performance degradation of non-real-time applications depending on the station-wide input limit, the effect of traffic smoothing on the performance of TCP sessions.

Another issue we are interested in is where to implement the traffic-smoothing function. Aside from putting a middleware under the IP layer, one can place the middleware between application layer and TCP/IP layer. The performance of this approach is likely to be poorer than the approach taken in this paper because the TCP/IP layer and lower layers will distort the traffic pattern smoothed by the middleware. However, this approach may be more cost-effective since the middleware is located closer to application programs. We are planning to compare this approach with the approach of this paper in terms of implementation cost and performance.

Acknowledgment

The authors thank Gary Workman for his support and technical comments on the empirical study of this paper.

References

[1] C. M. Aras, J. Kurose, D. S. Reeves, and H. Schulzrinne, "Real-time communication in packet-switched networks," *Proceedings of the IEEE*, vol. 82, pp. 122–139, Jan. 1994.

[2] Y. Shimokawa and Y. Shiobara, "Real-time Ethernet for industrial applications," in *Proc. of IECON*, pp. 829–834, 1985.

[3] D. W. Pritty, J. R. Malone, S. K. Banerjee, and N. L. Lawrie, "A real-time upgrade for Ethernet based factory networking," in *Proc. of IECON*, pp. 1631–1637, 1995.

[4] R. Court, "Real-time Ethernet," *Computer Communications*, vol. 15, pp. 198–201, Apr. 1992.

[5] C. Venkatramani and T. Chiueh, "Supporting real-time traffic on Ethernet," in *Proc. of Real-Time Systems Symposium*, pp. 282–286, Dec. 1994.

[6] S. L. Beuerman and E. Coyle, "The delay characteristics of CSMA/CD networks," *IEEE Trans. on Commun.*, vol. 36, pp. 553–563, May 1988.

[7] C. C. Chou and K. G. Shin, "Statistical real-time channels on multiaccess networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 8, pp. 769–780, Aug. 1997.

[8] T. Vo-Dai, "Steady-state analysis of CSMA-CD," in *Performance*, pp. 243–265, 1984.

[9] H. Zhang and E. Knightly, "Providing end-to-end statistical guarantees using bounding interval dependent stochastic models," in *Proc. of ACM SIGMETRICS*, pp. 211–220, 1994.

[10] L. Kleinrock, "Packet Switching in radio channels: part I-Carrier Sense Multiple-Access Modes and their throughput and delay characteristics," *IEEE Trans. on Communications*, vol. 23, pp. 1400–1416, Dec. 1975.

[11] R. L. Cruz, "A calculus for network delay, part I: network elements in isolation," *IEEE Trans. on Information Theory*, vol. 37, pp. 114–131, Jan. 1991.